05-30 00   *A*

<table>
<tr><td>

UTILITY PATENT APPLICATION TRANSMITTAL

(New Nonprovisional Applications Under 37 CFR § 1.53(b))

</td><td>

Attorney Docket No.
__002.0132.01__

</td></tr>
</table>

TO THE ASSISTANT COMMISSIONER FOR PATENTS:

Transmitted herewith is the patent application of ( ) application identifier or (X) first named inventor, <u>Victor Kouznetsov,</u> entitled <u>System And Method For Dynamically Detecting Computer Viruses Through Associative Behavioral Analysis Of Runtime State</u>, for a(n):

(X) Original Patent Application.

( ) Continuing Application (prior application not abandoned):
      ( ) Continuation   ( ) Divisional   ( ) Continuation-in-part (CIP)
      of prior application No: _____ Filed on: _____ .
      ( ) A statement claiming priority under 35 USC § 120 has been added to the specification.

Enclosed are:
      (X) Specification; __17__ Total Pages.   (X) Drawing(s); __8__ Total Sheets.
      (X) Oath or Declaration:
          (X) A Newly Executed Combined Declaration and Power of Attorney:
              (X) Signed.   ( ) Unsigned.   ( ) Partially Signed.
          ( ) A Copy from a Prior Application for Continuation/Divisional (37 CFR § 1.63(d)).
              ( ) Incorporation by Reference. The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied, is considered as being part of the disclosure of the accompanying application and is hereby incorporated herein by reference.
              ( ) Signed Statement Deleting Inventor(s) Named in the Prior Application. (37 CFR § 163(d)(2)).
      ( ) Power of Attorney.   (X) Return Receipt Postcard.
      ( ) Associate Power of Attorney.   (X) A Check in the amount of $708.00 for the Filing Fee.
      ( ) Preliminary Amendment.   ( ) Information Disclosure Statement and Form PTO-1449.
      (X) A Duplicate Copy of this Form for Processing Fee Against Deposit Account.
      ( ) A Certified Copy of Priority Documents (if foreign priority is claimed).
      ( ) Statement(s) of Status as a Small Entity.
      ( ) Statement(s) of Status as a Small Entity Filed in Prior Application, Status Still Proper and Desired.
      ( ) Other: _____ .

CLAIMS AS FILED				
FOR	NO. FILED	NO. EXTRA	RATE	FEE
Total Claims	21	1	$18.00	$ 18.00
Independent Claims	3	0	$78.00	$ 0.00
Multiple Dependent Claims (if applicable)				$0.00
Assignment Recording Fee				$0.00
Basic Filing Fee				$690.00
			Total Filing Fee	$ 708.00

Charge $_____ to Deposit Account __501144__ pursuant to 37 CFR § 1.25. At any time during the pendency of this application, please charge any fees required or credit any overpayment to this Deposit Account.

Respectfully submitted,

By: _____
    Patrick J.S. Inouye, Esq, Reg. No. 40297

Date: May 26, 2000

Correspondence Address: **22895**

Patrick J.S. Inouye, P.S.   **PATENT TRADEMARK OFFICE**
816 Second Avenue P.O. Box 21808
Seattle, WA 98111-3808
Phone: (206) 381-3900
Fax: (206) 381-3999

5

Patent Application

Docket No. 002.0132.01

NAI Docket No. 99.053.01

SYSTEM AND METHOD FOR DYNAMICALLY DETECTING

10   # COMPUTER VIRUSES THROUGH ASSOCIATIVE BEHAVIORAL

ANALYSIS OF RUNTIME STATE

Cross-Reference to Related Application

This patent application is related to a commonly-assigned U.S. patent application Serial No. ____, entitled "System And Method For Dynamically

15   Detecting Computer Viruses Through Behavioral Analysis Of Distributed Runtime State Using An Analysis Tree," filed May 26, 2000, pending, the disclosure of which is incorporated herein by reference.

Field of the Invention

The present invention relates in general to computer virus detection, and,

20   in particular, to a system and method for dynamically detecting computer viruses through associative behavioral analysis of runtime state.

Background of the Invention

Information networks interconnecting computational resources have become a mainstay of corporate enterprise computing environments. Generally,

25   these environments consist of host computer systems interconnected internally over an intranetwork to which individual workstations and network resources are connected. These intranetworks, also known as local area networks, make legacy databases and information resources widely available for access and data

exchange. These corporate resources can also be interconnected to wide area networks, including public information internetworks, such as the Internet, to enable internal users access to remote data exchange and computational resources, such as the World Wide Web, and to allow outside users access to select

5   corporate resources for completing limited transactions or data transfer.

Unfortunately, a minority of surreptitious users abuse computer interconnectivity by creating and distributing computer viruses and other types of unsolicited "malware." Computer viruses are self-replicating program code which often carry malicious and sometimes destructive payloads. Besides

10   computer viruses, malware can include Trojan horses, hoaxes, and "Spam" mail," such as, described in "McAfee.com: Virus Glossary of Terms," http://www.mcafee.com/viruses/virus_glossary.asp, Network Associates, Inc., (2000), the disclosure of which is incorporated herein by reference. Broadly stated, computer viruses include any type of object within which self-replicating

15   executable computer code can be stored, disseminated and directly or indirectly executed. These objects include conventional applications, macros, electronic mail messages and attachments, and even hypertext references.

In the prior art, there are three basic methodologies used to detect computer viruses. First, scanning performs a static check of stored computer files

20   and involves searching the files for byte sequences, known as virus signatures, which uniquely identify known viruses. However, scanning is time-consuming and can result in false positives, that is, non-viral files falsely identified as containing a virus. Moreover, the mere presence of a virus signature is no guarantee that the rest of the file is infected, as the "viral" code might never

25   actually execute due to program errors. Finally, polymorphic viruses can successfully evade detection virus scanners by changing their appearance with each replication.

Second, behavior checkers observe program execution and generate alerts upon the occurrence of events that indicate unusual program behavior. Such

30   behavior includes attempting to write to the boot sector of a storage drive or to

open executable files in read/write mode. However, behavior checkers typically focus on specific events occurring in isolation. Often, these events are being performed by legitimate applications and not by computer viruses. As well, the nature of these events is highly technical and suspect behavior alerts, while well

5   intentioned, might simply be ignored by the average user.

Third, integrity checkers create a log of characteristics for each of the stored computer files. The files are periodically re-checked by the integrity checker and the user is notified of any changes. However, some computer viruses falsify the characteristics of those files that have been infected, thereby defeating

10   the integrity checkers. Also, a change in characteristics does not necessary indicate the presence of a virus.

Therefore, there is a need for an approach to dynamically analyzing program behavior to determine if a computer virus is present. Preferably, knowledge of specific, pre-identified computer viruses would not be necessary

15   and the approach would instead observe behavioral patterns typical of computer viruses. The approach would track patterns of suspect events and analyze the stored patterns for repeated behaviors indicative of computer viruses.

Summary of the Invention

The present invention provides a system and method for dynamically

20   detecting computer virus activities through active monitoring of runtime execution state. The runtime state is monitored and compared to a set of monitored events. The occurrence of any one of the monitored events triggers tracking of the subsequent sequence of events. Suspect sequences characteristic of potentially viral activity are identified and histograms are generated. A virus

25   alert is generated if the histograms illustrate repeated suspect sequences.

An embodiment of the present invention is a system and a method for dynamically detecting computer viruses through associative behavioral analysis of runtime state. A group of monitored events is defined. Each monitored event includes a set of one or more actions defined within an object. Each action is

30   performed by one or more applications executing within a defined computing

environment. The runtime state within the defined computing environment is continuously monitored for an occurrence of any one of the monitored events in the group. The sequence of the execution of the monitored events is tracked for each of the applications. Each occurrence of a specific event sequence

5   characteristic of computer virus behavior and the application that performed the specific event sequence, are identified. A histogram describing the specific event sequence occurrence for each of the applications is created. Repetitions of the histogram associated with at least one object are identified.

One benefit of the present invention is that *a priori* knowledge of specific

10   computer viruses is not necessary. Rather, patterns of behavior characteristic of computer viruses are observed and analyzed. Repetitions of suspicious behavior generate an alert of potential viral activity.

Another benefit is the present invention can combine conventional approaches to dynamic virus checking, in particular, behavior checking, with

15   associative analyses using histograms.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the

20   invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

25   **Brief Description of the Drawings**

FIGURE 1 is a block diagram showing a distributed computing environment including a system for dynamically detecting computer viruses through associative behavioral analysis of runtime state in accordance with the present invention;

FIGURE 2 is a block diagram showing a detailed view of a client system for dynamically detecting computer viruses for use in the system of FIGURE 1;

FIGURE 3 is a block diagram showing the functional software modules of the monitor/analyzer of FIGURE 2;

FIGURE 4 is a data structure diagram showing an event log record used in the database of FIGURE 2;

FIGURE 5 is a process diagram showing the flow of requests and actions through the monitor/analyzer of FIGURE 3;

FIGURE 6 is a flow diagram showing a method for dynamically detecting computer viruses through associative behavioral analysis of runtime state in accordance with the present invention;

FIGURE 7 is a flow diagram showing the routine for collecting suspicious event data for use in the method of FIGURE 6; and

FIGURE 8 is a flow diagram showing the routine for analyzing suspicious event data for use in the method of FIGURE 6.

Detailed Description

FIGURE 1 is a block diagram showing a distributed computing environment 10 including a system for dynamically detecting computer viruses through associative behavioral analysis of runtime state 11 in accordance with the present invention. A client system 11 and a plurality of peer client systems 12 receive services from a local server 13 over a local area intranetwork 14. These services include standard network host services, including serving Web pages, hosting electronic mail services, managing files, information exchange, and the like. Other services and host configurations are feasible. In addition, a remote server 15 can be accessed via a wide area internetwork 16 that is interconnected with the intranetwork 14 via a gateway 17 or similar interfacing device. The client system 11 includes a monitor/analyzer 19 for dynamically detecting computer viruses, as further described below with reference to FIGURE 3, and a plurality of application programs 18, executing in a conventional manner.

The individual computer systems 11, 12, 13, 15 are general purpose, programmed digital computing devices consisting of a central processing unit (CPU), random access memory (RAM), non-volatile secondary storage, such as a hard drive or CD ROM drive, network interfaces, and peripheral devices,

5   including user interfacing means, such as a keyboard and display. Program code, including software programs, and data are loaded into the RAM for execution and processing by the CPU and results are generated for display, output, transmittal, or storage.

FIGURE 2 is a block diagram showing a detailed view 30 of a client

10   system 11 for dynamically detecting computer viruses for use in the system of FIGURE 1. The system 11 includes hardware 31 structured in a Von Neumann architecture, including a central processing unit (CPU), random access memory (RAM), storage device 36, and computational logic, as is conventional in the art. Resource allocation and control is managed by an operating system 32, such as

15   the Windows NT operating system, licensed by Microsoft Corporation, Redmond, Washington. A plurality of applications 33, 34, 35 are loaded into the RAM from storage device 36 and executed by the CPU.

The program state of the executing applications 33, 34, 35 is monitored by the monitor/analyzer 19 that generates histograms based on occurrences of

20   monitored events. The monitor/analyzer 19 functions as a logical "shim" interposed between the operating system 32 and each of the applications 33, 34, 35. Each system call is intercepted by the monitor/analyzer 19 which compares the requested system call to a list of monitored events. If the system call matches one of the monitored events, the monitor 19 determines whether the application is

25   performing a sequence of "suspicious" actions characteristic of computer viruses. If so, histograms of the event occurrences are generated and stored in a database 37 maintained in the storage device 36. Alternatively, the histograms could be stored in a centralized database for analysis of distributed runtime state, such as described in the related, commonly-assigned U.S. Patent application, entitled

30   "System And Method For Dynamically Detecting Computer Viruses Through

Behavioral Analysis Of Distributed Runtime State Using An Analysis Tree," filed May 26, 2000, pending, the disclosure of which is incorporated herein by reference. The histograms are analyzed to identify repetitions of suspect behavior.

5   FIGURE 3 is a block diagram showing the functional software modules 50 of the monitor/analyzer 19 of FIGURE 1. Each module is a computer program written as source code in a conventional programming language, such as the C or C++ programming languages, and is presented for execution by the CPU as object or byte code, as is known in the art. The various implementations of the source

10   code and object and byte codes can be held on a computer-readable storage medium or embodied on a transmission medium in a carrier wave.

The monitor/analyzer 19 consists of three components: collector 51, static analyzer 52, and dynamic analyzer 53. The collector 51 receives a list of events 54 to monitor. The collector 51 then continuously monitors the state of the

15   executing applications 33, 34, 35 for an occurrence of the monitored events 54. System calls generated by the applications 33, 34, 35 are intercepted by the collector 51 and are forwarded to the operating system 32. System call responses are similarly intercepted and forwarded to the applications 33, 34, 35 as necessary. Those system calls which match one of the monitored events are

20   logged as event occurrences 55.

The set of event occurrences 55 are analyzed by the static analyzer 52 for patterns of "suspicious" events. The static analyzer 52 performs behavior checking and generates alerts 57 if unusual program behavior is observed. The static analyzer 52 also generates histograms 56 of the occurrences of suspect

25   patterns. The histograms 56 are analyzed by the dynamic analyzer 53 which identifies behavioral repetitions within the histograms 56 which indicate behavior characteristic of a computer virus. Repeated patterns of suspect behavior associated with two or more of the applications 33, 34, 35 indicate potentially viral activities. These patterns are reported as alerts 57.

The monitor/analyzer 19 operates in accordance with a sequence of process steps, as further described below beginning with reference to FIGURE 5.

FIGURE 4 is a data structure diagram showing an event log record 70 used in the database 37 of FIGURE 2. Each record 70 is organized into four fields. The process identifier (ID) 71 and application name 72 fields respectively store the process number and name of the application 33, 34, 35 (shown in FIGURE 2) to which the recorded monitored event is associated. Similarly, the event identifier (ID) 73 and event time 74 fields store the number of the monitored event and the time at which the monitored event occurred. The data stored in the event log records 70 is used to generate histograms of specific event sequences.

FIGURE 5 is a process diagram 90 showing the flow of requests and actions through the monitor/analyzer 19 of FIGURE 3. The monitor/analyzer 19 executing on the client 11 receives incoming system calls 91, 92, 93 generated by the applications 33, 34, 35 (shown in FIGURE 2) and forwards any system call responses 94, 95, 96 back to the requesting applications 33, 34, 35. Alerts 97 are generated as "suspect" behavior is observed by the monitor/analyzer 19.

In the described embodiment, the following categories of events are monitored:

(1) Files accesses, including read, write, redirection, open, close, and modify operations;

(2) Program executions;

(3) Direct disk accesses, particularly boot sector accesses;

(4) Media formatting operations;

(5) Sending of electronic mail;

(6) System configuration area accesses, such as Registry files;

(7) Changes to security settings;

(8) Impersonations; and

(9) System calls having the ability to monitor system input/output activities.

The foregoing list of monitored event categories is non-exclusive and other events could be included.

Specific sequences of the monitored events are organized into histograms which identify behavior characteristic of computer viruses. For example, a widely used virus replication technique infects by appending viral program code to the end of an application file. The replication sequence consists of four events, described as follows:

$event_{(1.1)}$: open application program file;

$event_{(1.2)}$: reposition file pointer to end of application program file;

$event_{(1.3)}$: write n bytes of new data; and

$event_{(1.4)}$: close application program file.

Here, the number of new data bytes n written is the key: a pattern of the same number of bytes n repeatedly written into different application program files at the end of the files indicates potentially viral activity.

An example of an electronic mail-based computer virus steals addresses from an address book, such as the Outlook 2000 application, licensed by Microsoft Corporation, Redmond, Washington, and propagates via the electronic mail system. The sequence consists of four events, described as follows:

$event_{(2.1)}$: open an address book for an electronic mail application;

$event_{(2.2)}$: create an electronic mail message with an attachment having m bytes;

$event_{(2.3)}$: repeating $event_{(2.2)}$ for one or more of the names in the address book; and

$event_{(2.4)}$: closing the address book.

Here, the number of bytes in the attachment m can be tracked and a repeated pattern would indicate potential replication.

An example of an "impersonation" occurs when a virus program changes file characteristics to match those of a privileged and typically well-known systems program. The sequence consists of three events, described as follows:

$event_{(3.1)}$: launch a privileged application;

event(3.2): query the security attributes, permissions, and other file

characteristics of the privileged application; and

event(3.3): modify the security attributes, permissions, and other file

characteristics of the virus program to match those of the

5   privileged application.

Finally, an example of illicit input/output monitoring activity includes a

sequence consisting of two events, described as follows:

event(4.1): launch the illicit monitoring application; and

event(4.2): register the illicit monitoring application with the operating

10   system as an input/output system monitor.

Still other sequences characteristic of computer virus behaviors are

described in M. Ludwig, "The Giant Black Book of Computer Viruses," Part I,

pp. 27-271, Am. Eagle Pub., Inc. (2d ed. 1998), the disclosure of which is

incorporated herein by reference.

15   FIGURE 6 is a flow diagram showing a method for dynamically detecting

computer viruses through associative behavioral analysis of runtime state 110 in

accordance with the present invention. First, the monitor/analyzer 19 is initialized

(block 111) by loading the monitored events form the events list 54 (shown in

FIGURE 3). Execution then continues in three substantially concurrent threads

20   which are iteratively processed (blocks 112-116). In the first thread, event data

consisting of observed occurrences of monitored events is collected (block 113),

as further described below with reference to FIGURE 7. In the second thread, the

event data is analyzed (block 114), as further described below with reference to

FIGURE 8. Finally, in the third thread, the event data is statically analyzed for

25   patterns of "suspicious" events (block 115) using behavioral checking, such as

described in M. Ludwig, "The Giant Black Book of Computer Viruses," Ch. 23,

pp. 273-280, Am. Eagle Pub., Inc. (2d ed. 1998), the disclosure of which is

incorporated herein by reference. The execution threads continue execution until

the program is terminated.

FIGURE 7 is a flow diagram showing the routine for collecting suspicious event data 130 for use in the method of FIGURE 6. The purpose of this routine is to collect data for those system calls 91, 92, 93 (shown in FIGURE 5) that match one of the monitored events in the events list 54 (shown in FIGURE 3). Thus,

5   system calls are monitored (block 131). If the monitored system call matches a triggering event (block 132), an event log record 70 (shown in FIGURE 4) is created and the event is stored in the database 37 (block 133). If there are more system calls to be monitored (block 134), processing continues as before (block 131). Otherwise, the routine returns.

10   FIGURE 8 is a flow diagram showing the routine for analyzing suspicious event data 150 for use in the method of FIGURE 6. The purpose of this routine is to analyze the stored monitored events in the database 37 (shown in FIGURE 2) by generating histograms 56 (shown in FIGURE 3) and identifying repetitions of suspicious behavioral patterns. First, the records for the monitored events 70 are

15   retrieved for each of the applications 33, 34, 35 (block 151). Next, each of the retrieved events are analyzed in an iterative processing loop (blocks 152-156) as follows. First, a histogram of event occurrences is created (block 153). If the histogram indicates a history of "suspicious" activities (block 154), the monitored event is flagged and an alert is generated (block 155). If there are more records in

20   the database (block 157), processing continues as before (block 151). Otherwise, the routine returns.

While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without

25   departing from the spirit and scope of the invention.

What is claimed is:

1 1. A system for dynamically detecting computer viruses through

2 associative behavioral analysis of runtime state, comprising:

3   a parameter set stored on a client system defining a group of monitored

4 events which each comprise a set of one or more actions defined within an object,

5 each action being performed by one or more applications executing within a

6 defined computing environment;

7   a monitor executing on the client system, comprising:

8   a collector continuously monitoring the runtime state within the

9 defined computing environment for an occurrence of any one of the monitored

10 events in the group and tracking the sequence of the execution of the monitored

11 events for each of the applications; and

12   an analyzer identifying each occurrence of a specific event

13 sequence characteristic of computer virus behavior and the application which

14 performed the specific event sequence, creating a histogram describing the

15 specific event sequence occurrence for each of the applications, and identifying

16 repetitions of the histogram associated with at least one object.

1 2. A system according to Claim 1, further comprising:

2   a storage manager organizing the histograms into plurality of records

3 ordered by object, application, and monitored event.

1 3. A system according to Claim 2, further comprising:

2   a structured database in which the plurality of records is stored; and

3   the storage manager storing each histogram for each such specific event

4 sequence occurrence in one such database record identified by the application by

5 which the specific event sequence was performed.

1 4. A system according to Claim 3, further comprising:

2   the storage manager configuring the structured database as an event log

3 organized by each event in the group of monitored events and updating the

4   database record storing each specific event sequence occurrence with a revised

5   histogram as each such occurrence is identified.

1   5.   A system according to Claim 1, further comprising:

2   the analyzer detecting suspect activities within each histogram, each

3   suspect activity comprising a set of known actions comprising a computer virus

4   signature.

1   6.   A system according to Claim 6, wherein each such suspect activity

2   is selected from the class of actions comprising file accesses, program executions,

3   message transmissions, configuration area accesses, security setting accesses, and

4   impersonations.

1   7.   A system according to Claim 6, wherein each such suspect activity

2   is selected from the group comprising files accesses, program executions, direct

3   disk accesses, media formatting operations, sending of electronic mail, system

4   configuration area accesses, changes to security settings, impersonations, and

5   system calls having the ability to monitor system input/output activities.

1   8.   A system according to Claim 1, wherein the computer virus

2   comprises at least one form of unauthorized content selected from the group

3   comprising a computer virus application, a Trojan horse application, and a hoax

4   application.

1   9.   A method for dynamically detecting computer viruses through

2   associative behavioral analysis of runtime state, comprising:

3   defining a group of monitored events which each comprise a set of one or

4   more actions defined within an object, each action being performed by one or

5   more applications executing within a defined computing environment;

6   continuously monitoring the runtime state within the defined computing

7   environment for an occurrence of any one of the monitored events in the group;

8   tracking the sequence of the execution of the monitored events for each of

9   the applications;

10   identifying each occurrence of a specific event sequence characteristic of

11   computer virus behavior and the application which performed the specific event

12   sequence;

13   creating a histogram describing the specific event sequence occurrence for

14   each of the applications; and

15   identifying repetitions of the histogram associated with at least one object.

1   10.   A method according to Claim 9, further comprising:

2   organizing the histograms into plurality of records ordered by object,

3   application, and monitored event.

1   11.   A method according to Claim 10, further comprising:

2   maintaining a structured database in which the plurality of records is

3   stored; and

4   storing each histogram for each such specific event sequence occurrence

5   in one such database record identified by the application by which the specific

6   event sequence was performed.

1   12.   A method according to Claim 11, further comprising:

2   configuring the structured database as an event log organized by each

3   event in the group of monitored events; and

4   updating the database record storing each specific event sequence

5   occurrence with a revised histogram as each such occurrence is identified.

1   13.   A method according to Claim 9, further comprising:

2   detecting suspect activities within each histogram, each suspect activity

3   comprising a set of known actions comprising a computer virus signature.

1   14.   A method according to Claim 13, wherein each such suspect

2   activity is selected from the class of actions comprising file accesses, program

3   executions, message transmissions, configuration area accesses, security setting

4   accesses, and impersonations.

1   15.   A method according to Claim 13, wherein each such suspect

2   activity is selected from the group comprising files accesses, program executions,

3   direct disk accesses, media formatting operations, sending of electronic mail,

4   system configuration area accesses, changes to security settings, impersonations,

5   and system calls having the ability to monitor system input/output activities.

1   16.   A method according to Claim 9, wherein the computer virus

2   comprises at least one form of unauthorized content selected from the group

3   comprising a computer virus application, a Trojan horse application, and a hoax

4   application.

1   17.   A computer-readable storage medium holding code for

2   dynamically detecting computer viruses through associative behavioral analysis of

3   runtime state, comprising:

4   defining a group of monitored events which each comprise a set of one or

5   more actions defined within an object, each action being performed by one or

6   more applications executing within a defined computing environment;

7   continuously monitoring the runtime state within the defined computing

8   environment for an occurrence of any one of the monitored events in the group;

9   tracking the sequence of the execution of the monitored events for each of

10   the applications;

11   identifying each occurrence of a specific event sequence characteristic of

12   computer virus behavior and the application which performed the specific event

13   sequence;

14   creating a histogram describing the specific event sequence occurrence for

15   each of the applications; and

16   identifying repetitions of the histogram associated with at least one object.

1   18.   A storage medium according to Claim 17, further comprising:

2   organizing the histograms into plurality of records ordered by object,

3   application, and monitored event.

1   19.   A storage medium according to Claim 18, further comprising:

2   maintaining a structured database in which the plurality of records is

3   stored; and

4   storing each histogram for each such specific event sequence occurrence

5   in one such database record identified by the application by which the specific

6   event sequence was performed.

1   20.   A storage medium according to Claim 19, further comprising:

2   configuring the structured database as an event log organized by each

3   event in the group of monitored events; and

4   updating the database record storing each specific event sequence

5   occurrence with a revised histogram as each such occurrence is identified.

1   21.   A storage medium according to Claim 17, further comprising:

2   detecting suspect activities within each histogram, each suspect activity

3   comprising a set of known actions comprising a computer virus signature.

SYSTEM AND METHOD FOR DYNAMICALLY DETECTING COMPUTER VIRUSES THROUGH ASSOCIATIVE BEHAVIORAL ANALYSIS OF RUNTIME STATE

Abstract

5   A system and a method for dynamically detecting computer viruses through associative behavioral analysis of runtime state are described. A group of monitored events is defined. Each monitored event includes a set of one or more actions defined within an object. Each action is performed by one or more applications executing within a defined computing environment. The runtime

10   state within the defined computing environment is continuously monitored for an occurrence of any one of the monitored events in the group. The sequence of the execution of the monitored events is tracked for each of the applications. Each occurrence of a specific event sequence characteristic of computer virus behavior and the application that performed the specific event sequence, are identified. A

15   histogram describing the specific event sequence occurrence for each of the applications is created. Repetitions of the histogram associated with at least one object are identified.

Figure 1.

10

Figure 2.

Client 11

30

Figure 3.

50

19

51

53

52

33 Application₁

34 Application₂

35 Applicationₙ

32 Operating System

Monitor/Analyzer

Collector

Static Analyzer

Dynamic Analyzer

Events List 54

Events Occurrences 55

Histograms 56

Alerts 57

Figure 4.

70
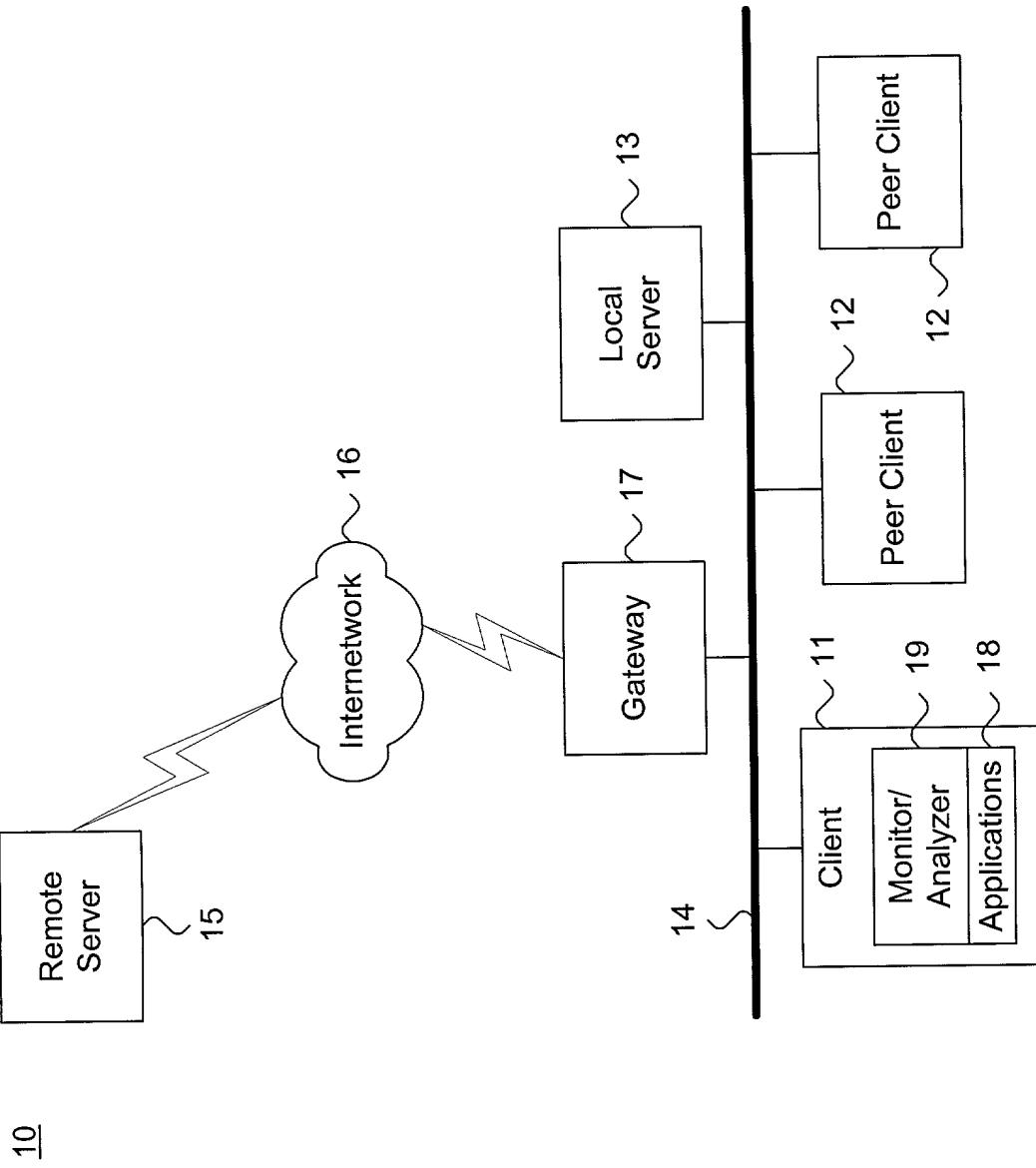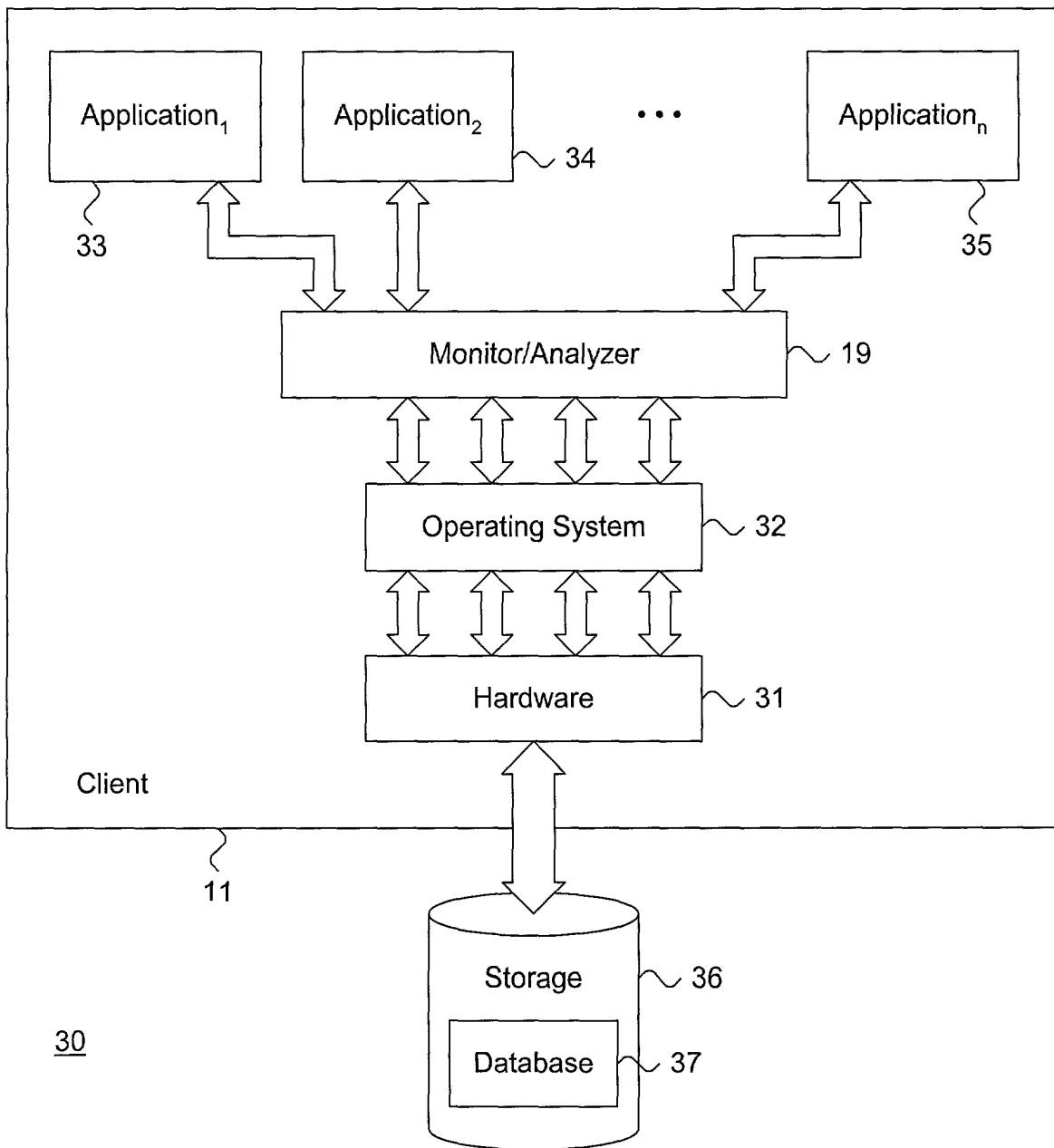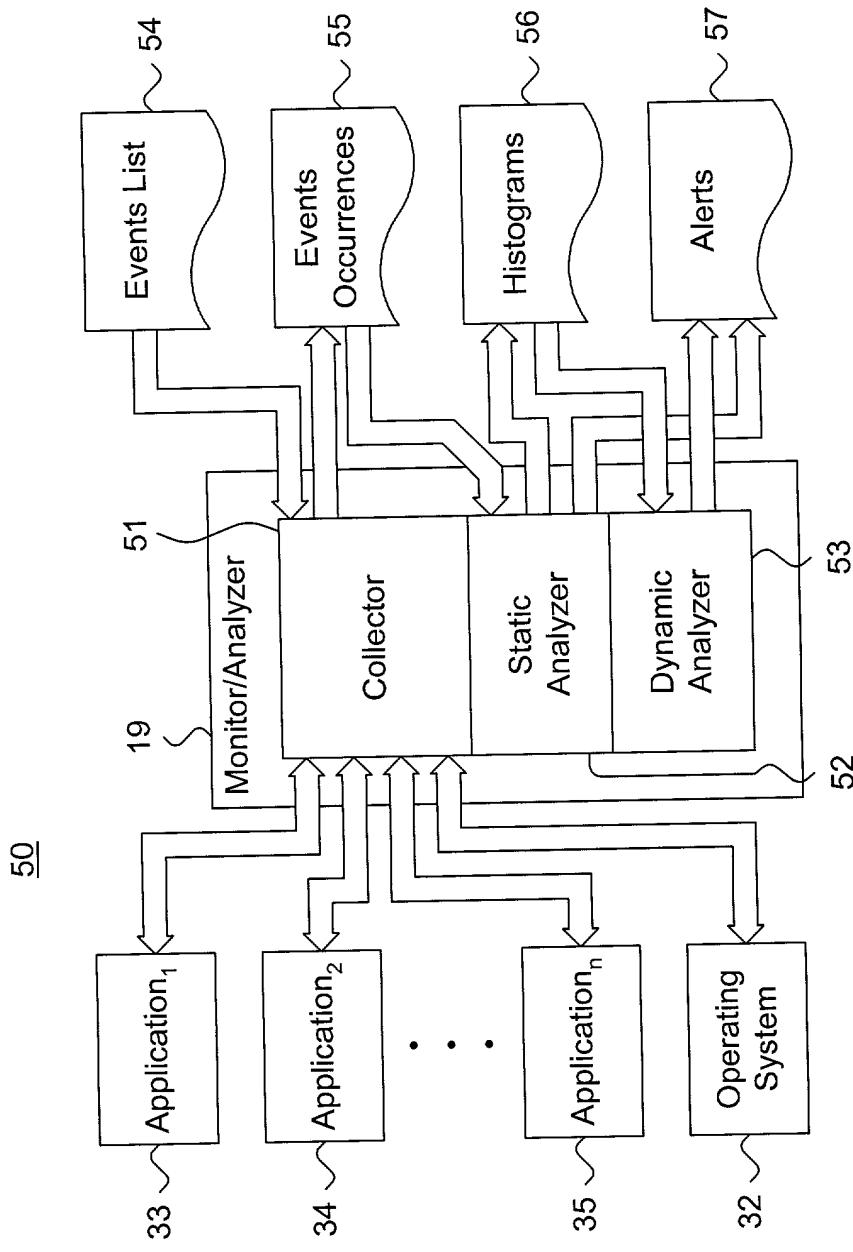
Process ID	Application Name	Event ID	Event Time
71	72	73	74

Figure 5.

90

System
Call₁ — 91

System
Call₂ — 92

System
Callₙ — 93

System
Call₁ — 94

System
Call₂ — 95

System
Callₙ — 96

Monitor/Analyzer

19

Alerts — 97

Figure 6.

```
                    ╭─────────────╮
                    │    Start    │ ～110
                    ╰─────────────╯
                           │
                           ▼
                 ┌─────────────────┐
                 │   Initialize    │ ～111
                 └─────────────────┘
                           │
                           ▼
                 ╱───────────────────╲
                ╱     Do while        ╲ ～112
               ╱───────────────────────╲
                           │
          ┌────────────────┼────────────────────┐
          ▼                ▼                     ▼
  ┌──────────────┐  ┌──────────────┐    ┌────────────────┐
113～│ Collect event│  │ Analyze event│    │ Static analysis│ ～115
  │    data      │  │    data      │    │                │
  └──────────────┘  └──────────────┘    └────────────────┘
          │            114              │
          ▼                ▼                     │
          └────────────────┤─────────────────────┘
                           ▼
                 ╲───────────────────╱
                  ╲     End do       ╱ ～116
                   ╲───────────────╱
                           │
                           ▼
                    ╭─────────────╮
                    │     End     │
                    ╰─────────────╯
```

Figure 7.

```
        ┌─────────────┐
        │ Collect Event│ ⌇130
        │    Data     │
        └──────┬──────┘
               │          ┌──────────────┐
               ▼          │              │
        ┌─────────────┐   │              │
        │ Monitor system│ ⌇131           │
        │    calls     │   │             │
        └──────┬──────┘   │              │
               │          │              │
               ▼          │              │
         N   ╱────────╲  ⌇132            │
        ◄───╱Triggering ╲                │
            ╲  event?   ╱                │
         │   ╲────────╱                  │
         │        │ Y                    │
         │        ▼                      │
         │  ┌─────────────┐              │
         │  │Create record │ ⌇133        │
         │  │of source     │             │
         │  │application and│            │
         │  │event in      │             │
         │  │database      │             │
         │  └──────┬──────┘              │
         │         │                     │
         └────────►│                     │
                   ▼                     │
          134  ╱────────╲    Y           │
              ╱  More?   ╲───────────────┘
              ╲         ╱
               ╲───────╱
                   │ N
                   ▼
            ┌─────────────┐
            │   Return    │
            └─────────────┘
```
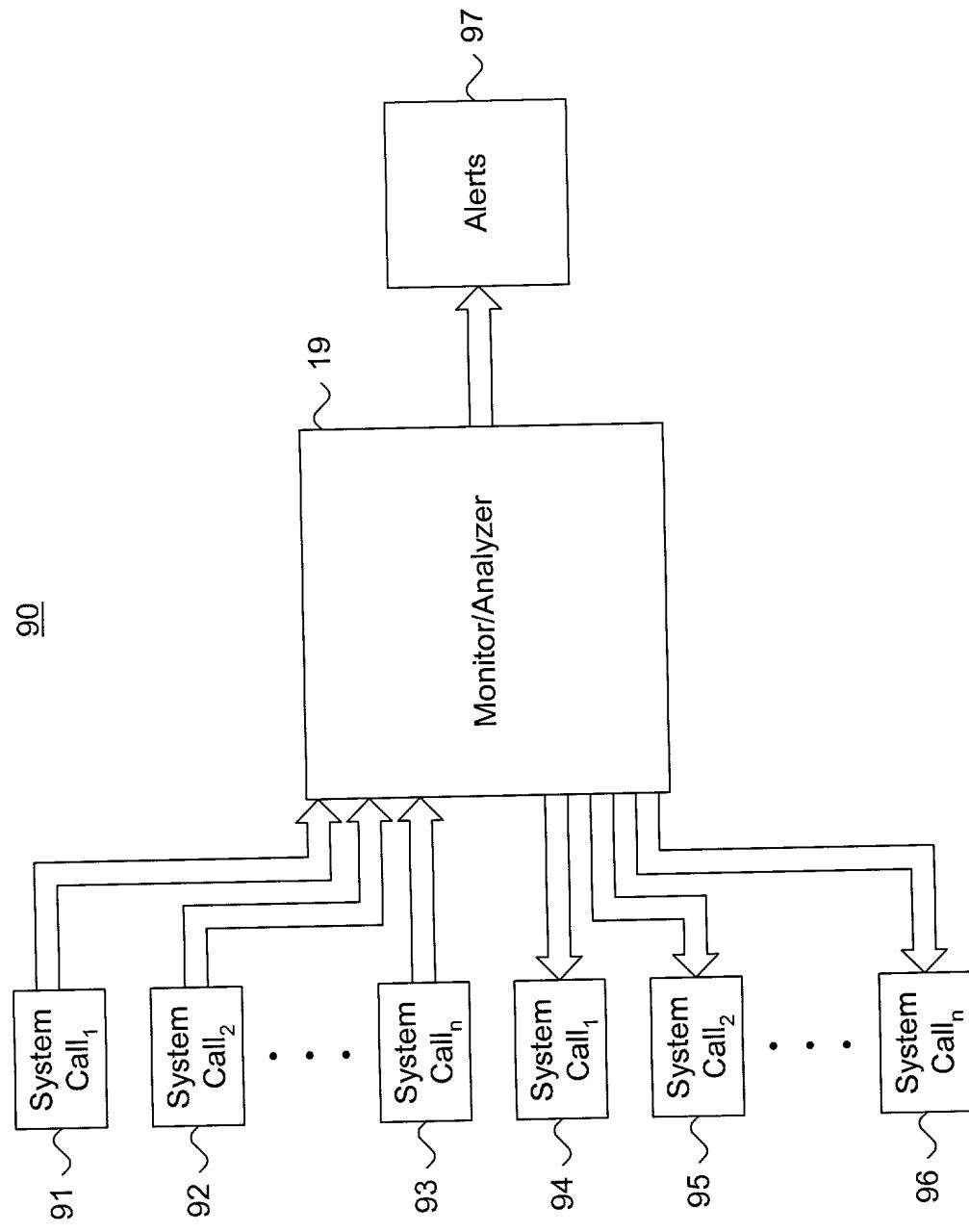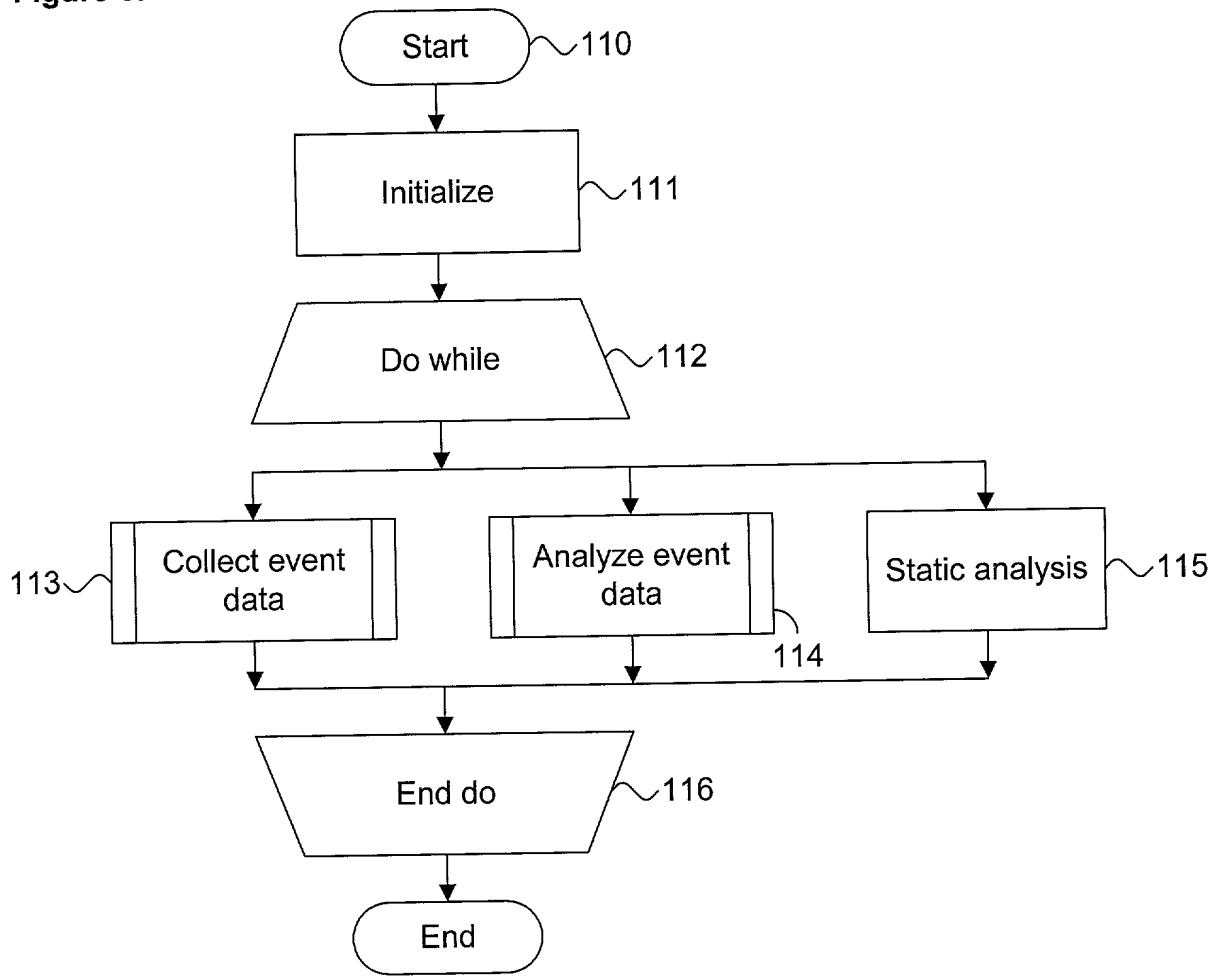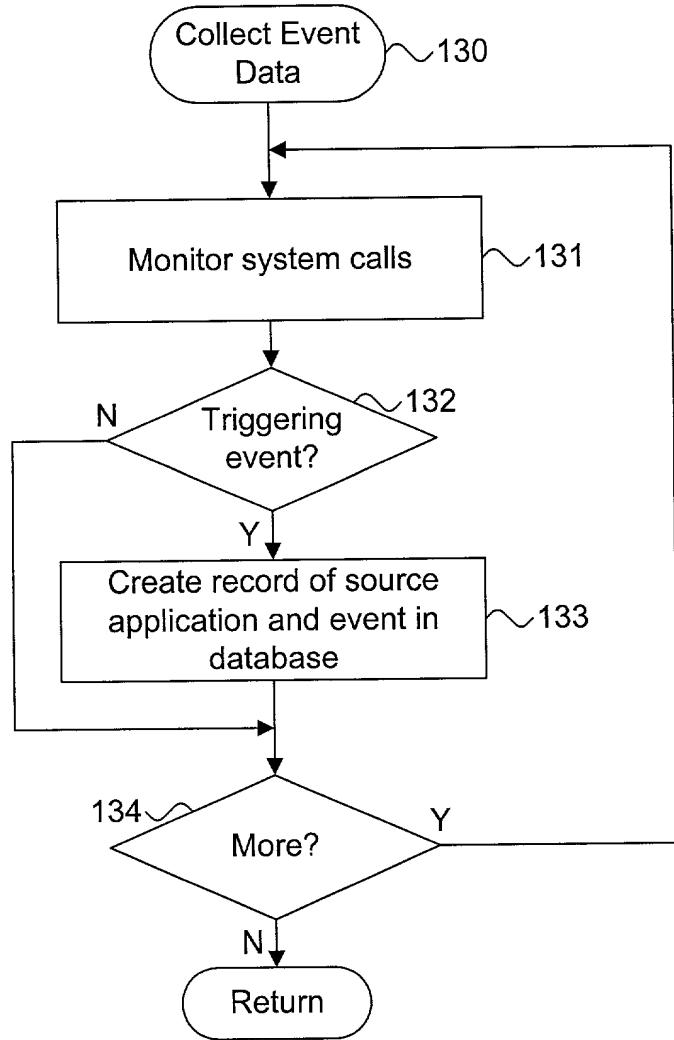
Figure 8.
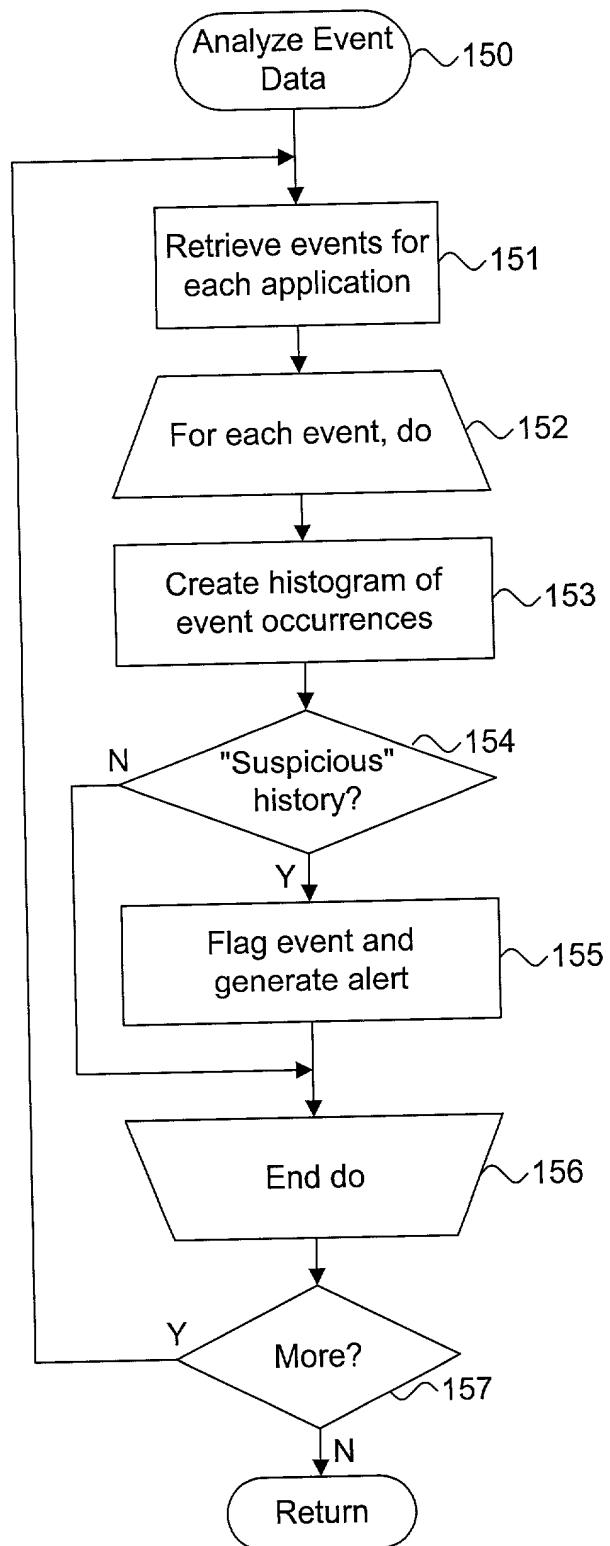
PATENT APPLICATION

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION	ATTORNEY DOCKET NO. 002.0132.01

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

<u>System And Method For Dynamically Detecting Computer Viruses Through Associative Behavioral Analysis Of Runtime State</u>

the specification of which is attached hereto unless the following box is checked:

() was filed on _____ as US Application Serial No. or PCT International Application

Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefit(s) under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
			YES:___ NO:___
			YES:___ NO:___

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS(patented/pending/abandoned)

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) listed below to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Patrick J.S. Inouye, Esq, Reg. No. 40297   **Brian J. Daiuto, Esq, Reg. No. 38422**

Send Correspondence to: Patrick J.S. Inouye, Esq Patrick J.S. Inouye, P.S. 816 Second Avenue P.O. Box 21808 Seattle, WA 98111-3808	Direct Telephone Calls To: Patrick J.S. Inouye, Esq (206) 381-3900

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: <u>Victor Kouznetsov</u>   Citizenship: <u>Russia</u>

Residence: <u>20287 SW Tremont Way, Aloha, Oregon 97007</u>

Post Office Address: <u>Same</u>

<u>Victor Kouznetsov</u>   <u>5/24/00</u>

Inventor's Signature   **Date**